REMARKS

Claims 1, 3-16, 20, and 36-48 are pending in this application. Claims 1, 5, 7-12, and 36-38 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over "HTML 4.0 Specification" (hereinafter "HTML") in view of U.S. Patent No. 5,974,416 to Anand et al. (hereinafter "Anand"). Claims 3 and 4 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over HTML in view of Anand and further in view of "Hyperactions in a Markup Language." Claims 13-15 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over HTML in view of Anand and further in view of "Distributed Databases." Claim 16 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over HTML in view of Anand and further in view of "Distributed Databases" and "Module mod_log_common." Claims 6 and 20 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over HTML in view of Anand and further in view of U.S. Patent No. 5,826,034 to Albal.

By the foregoing amendments, Applicant has added new claims 39-48, including new independent claims 39, 40, and 46. New independent claim 39 more specifically defines the types of information stored in the data table and the actions associated with the stored information. New independent claim 40 more specifically defines the functionality of the search unit. New independent claim 46 more

specifically defines the functionality of the search unit based on how the accessed information is structured.

HTML discloses the building blocks for any HTML-based document, including such items as tables having columns and rows.

Anand discloses a method and tabular data stream format "for marshaling tabular data for transfer between clients and servers" (column 2, lines 3-12). In the method of Anand, a user queries a remote database via a client process. This query is sent to a Web server, which queries databases via a database interface (column 4, line 62 to column 5, line 3; column 5, lines 46-53).

The information is retrieved from the databases and is sent back to the user, via an Advanced Data TableGram (ADTG) format. The user manipulates a local copy of the retrieved data. If the user makes changes to the data, the changes are uploaded to the databases via the ADTG format. (See column 7, line 45 to column 8, line 3.)

While the method and data format of Anand permit a user to indirectly manipulate data stored in one or more remote databases, the disclosures of HTML and Anand, taken together, do not disclose the features of the present invention. In particular, the cited references do not disclose a type of search that matches information in a key phrase field and in each of a plurality of column headings, as recited in independent claim 1 of the present invention. Anand contains no mention

of the types of queries that are performed; the key feature disclosed in Anand is that a query returns data in a tabular format, according to the ADTG format defined in the specification. There is no hint or suggestion in Anand that would lead one skilled in the art to create a query that is a combination of a key phrase field and a column heading as recited in the present invention.

In the Advisory Action, the Examiner cited additional sections of Anand (i.e., column 6 lines 49-59 and column 19, lines 5-25), referring to the SQL Join command. Anand repackages the information into rowsets for easier transmission, and as noted at column 19, lines 12-14, "[t]he rowsets may be composed of columns belonging to different base tables (created using SQL joins)."

Even apart from its use in Anand, the SQL Join query does not lead one skilled in the art to the present invention as defined in the independent claims. A Join query is used to select data from two tables based on a primary key, in order to retrieve a complete query result. (See the attached explanation of the SQL Join command, retrieved from http://www.w3schools.com on May 26, 2005.)

It is respectfully submitted that the remarks made herein place pending claims 1, 3-16, 20, and 36-48 in condition for allowance. Accordingly, entry of this amendment as well as reconsideration and allowance of pending claims 1, 3-16, 20, and 36-48 are respectfully requested.

If the Examiner does not believe that the claims are in condition for allowance, the Examiner is respectfully requested to contact the undersigned at 215-568-6400.

Respectfully submitted,

Richard B. Himmelstein

By _____
Steven J. Gelman
Registration No. 41,034
(215) 568-6400

Volpe and Koenig, P.C.
United Plaza, Suite 1600
30 South 17th Street
Philadelphia, PA 19103

SJG/slp
Enclosure

SQL Join

Previous Next

Joins and Keys

Sometimes we have to select data from two or more tables to make our result complete. We have to perform a join.

Tables in a database can be related to each other with keys. A primary key is a column with a unique value for each row. The purpose is to bind data together, across tables, without repeating all of the data in every table.

In the "Employees" table below, the "Employee_ID" column is the primary key, meaning that **no** two rows can have the same Employee_ID. The Employee_ID distinguishes two persons even if they have the same name.

When you look at the example tables below, notice that:

- The "Employee_ID" column is the primary key of the "Employees" table
- The "Prod_ID" column is the primary key of the "Orders" table
- The "Employee_ID" column in the "Orders" table is used to refer to the persons in the "Employees" table without using their names

---

Employees:

Employee_ID	Name
01	Hansen, Ola
02	Svendson, Tove
03	Svendson, Stephen
04	Pettersen, Kari

Orders:

Prod_ID	Product	Employee_ID
234	Printer	01
657	Table	03
865	Chair	03

Referring to Two Tables

We can select data from two tables by referring to two tables, like this:

Example

Who has ordered a product, and what did they order?

```
SELECT Employees.Name, Orders.Product
FROM Employees, Orders
WHERE Employees.Employee_ID=Orders.Employee_ID
```

Result

Name	Product
Hansen, Ola	Printer
Svendson, Stephen	Table
Svendson, Stephen	Chair

Example

Who ordered a printer?

```
SELECT Employees.Name
FROM Employees, Orders
WHERE Employees.Employee_ID=Orders.Employee_ID
AND Orders.Product='Printer'
```

Result

Name
Hansen, Ola

Using Joins

OR we can select data from two tables with the JOIN keyword, like this:

Example INNER JOIN

Syntax

```
SELECT field1, field2, field3
FROM first_table
INNER JOIN second_table
ON first_table.keyfield = second_table.foreign_keyfield
```

Who has ordered a product, and what did they order?

```
SELECT Employees.Name, Orders.Product
FROM Employees
INNER JOIN Orders
ON Employees.Employee_ID=Orders.Employee_ID
```

The INNER JOIN returns all rows from both tables where there is a match. If there are rows in Employees that do not have matches in Orders, those rows will **not** be listed.

Result

Name	Product
Hansen, Ola	Printer
Svendson, Stephen	Table
Svendson, Stephen	Chair

Example LEFT JOIN

Syntax

```
SELECT field1, field2, field3
FROM first_table
LEFT JOIN second_table
ON first_table.keyfield = second_table.foreign_keyfield
```

List all employees, and their orders - if any.

```
SELECT Employees.Name, Orders.Product
FROM Employees
LEFT JOIN Orders
ON Employees.Employee_ID=Orders.Employee_ID
```

The LEFT JOIN returns all the rows from the first table (Employees), even if there are no matches in the second table (Orders). If there are rows in Employees that do not have matches in Orders, those rows **also** will be listed.

Result

Name	Product
Hansen, Ola	Printer
Svendson, Tove	
Svendson, Stephen	Table
Svendson, Stephen	Chair
Pettersen, Kari	

Example RIGHT JOIN

Syntax

```
SELECT field1, field2, field3
FROM first_table
RIGHT JOIN second_table
ON first_table.keyfield = second_table.foreign_keyfield
```

List all orders, and who has ordered - if any.

```
SELECT Employees.Name, Orders.Product
FROM Employees
RIGHT JOIN Orders
ON Employees.Employee_ID=Orders.Employee_ID
```

The RIGHT JOIN returns all the rows from the second table (Orders), even if there are no matches in the first table (Employees). If there had been any rows in Orders that did not have matches in Employees, those rows **also** would have been listed.

Result

Name	Product
Hansen, Ola	Printer
Svendson, Stephen	Table
Svendson, Stephen	Chair

Example

Who ordered a printer?

```
SELECT Employees.Name
FROM Employees
INNER JOIN Orders
ON Employees.Employee_ID=Orders.Employee_ID
WHERE Orders.Product = 'Printer'
```

Result

Name
Hansen, Ola

Previous Next